In [4]:

```
# ============================================================
============================
# Assignment: Final Project - Phase 2
# Filename: final_project_2.py
# Description: Final Project
# Date: 04/19/2020
# Author: Tarini Dash
# ============================================================
============================

import pandas as pd
import numpy as np


def initial(read_df):
#     Thanks Benjamin for one liner.
    return read_df[read_df.columns[1:-1]].sample(2)
```

```
def get_cluster_value(read_df,index,mu_2,mu_4):
    dist_from_centroid_2 = np.sqrt(np.square(read_df[
read_df.columns[1:-1]].loc[index] - mu_2).sum())
    dist_from_centroid_4 = np.sqrt(np.square(read_df[
read_df.columns[1:-1]].loc[index] - mu_4).sum())
    if(dist_from_centroid_2 < dist_from_centroid_4):
        return 2
    if(dist_from_centroid_4 < dist_from_centroid_2):
        return 4


def recompute(cluster_2,cluster_4):
    return cluster_2[cluster_2.columns[1:-2]].mean(ax
is = 0), cluster_4[cluster_4.columns[1:-2]].mean(axis
= 0)



def assign_cluster(read_df,mu_2,mu_4):
    col1 = ["Scn", "A2", "A3", "A4", "A5", "A6", "A7"
```

```
, "A8", "A9", "A10", "Class","predicted_class"]
    predicted_clusters = pd.DataFrame()
    for i in range(len(read_df)):
#         data = pd.Series([read_df["Scn"].loc[i],
#                           read_df["A2"].loc[i],
#                           read_df["A3"].loc[i],
#                           read_df["A4"].loc[i],
#                           read_df["A5"].loc[i],
#                           read_df["A6"].loc[i],
#                           read_df["A7"].loc[i],
#                           read_df["A8"].loc[i],
#                           read_df["A9"].loc[i],
#                           read_df["A10"].loc[i],
#                           read_df["Class"].loc[i]
#                           ,get_cluster_value(read_df,i,mu_2,mu_4)], index = col1)
#         print(data)

        # not sure how to assign data from read_df + additional column predicted_class with [read_df.iloc[i], get_cluster_value(df,i,mu_2,mu_4)]. It does not
```

```
 work.
        predicted_clusters = predicted_clusters.appen
d(pd.Series([read_df["Scn"].loc[i],

read_df["A2"].loc[i],

read_df["A3"].loc[i],

read_df["A4"].loc[i],

read_df["A5"].loc[i],

read_df["A6"].loc[i],

read_df["A7"].loc[i],

read_df["A8"].loc[i],

read_df["A9"].loc[i],

read_df["A10"].loc[i],
```

```
read_df["Class"].loc[i]

,get_cluster_value(read_df,i,mu_2,mu_4)], index = col
1) ,ignore_index=True)


    return predicted_clusters



def loop_through(read_df,mu_2,mu_4):
    for i in range(1,51):
#         print("mu_2 :", mu_2)
#         print("mu_4 :", mu_4)
        predicted_clusters = assign_cluster(read_df,m
u_2,mu_4)

        cluster_2 = predicted_clusters.loc[predicted_
clusters['predicted_class'] == 2]
        cluster_4 = predicted_clusters.loc[predicted_
```

```
clusters['predicted_class'] == 4]

        new_mu_2, new_mu_4 = recompute(cluster_2,clus
ter_4)
        if(np.array_equal(new_mu_2, mu_2) & np.array_
equal(new_mu_4, mu_4)):
#             print("centroids matched with previous
 centroids. No change in centroids")
            return i, mu_2, mu_4, predicted_clusters
            break;
        else:
            mu_2 = new_mu_2
            mu_4 = new_mu_4
    return i, mu_2, mu_4, predicted_clusters



def main():
    #ref - https://pandas.pydata.org/pandas-docs/stab
le/reference/api/pandas.set_option.html
    pd.set_option('precision', 0)
```

```
    col = ["Scn", "A2", "A3", "A4", "A5", "A6", "A7",
"A8", "A9", "A10", "Class"]

    # read from file
    read_df = pd.read_csv('breast-cancer-wisconsin.da
ta', na_values = '?', names = col)

    # replace missing value with mean
    read_df = read_df.fillna(read_df.mean()['A7'])

    sample_df = initial(read_df)

    count = 2
    for index, row in sample_df.iterrows():
        print("Randomly selected row ",index, "for ce
ntroid mu_"+str(count), end="\n\n")
        print("Initial centroid mu_"+str(count)+":")
        print(row,end="\n\n")
        count += 2
```

```
    mu_2 , mu_4 = sample_df.values[0].astype(int), sa
mple_df.values[1].astype(int)

    i,final_mu_2,final_mu_4, predicted_clusters = loo
p_through(read_df,mu_2,mu_4)

    print("Program ended after ",i, " iterations.",en
d="\n\n")
    print("Final centroid mu_2:\n",final_mu_2,end="\n
\n")
    print("Final centroid mu_4:\n",final_mu_4,end="\n
\n")
    print("Final cluster assignment:",end="\n\n")
    predicted_clusters = predicted_clusters[['Scn',
'Class', 'predicted_class']]
#     print(predicted_clusters,end="\n\n")
#     print first twenty rows
    print(predicted_clusters.head(20),end="\n\n")

    cluster_2 = predicted_clusters.loc[predicted_clus
ters['predicted_class'] == 2]
```

```
    cluster_4 = predicted_clusters.loc[predicted_clus
ters['predicted_class'] == 4]
    print("Data points in Predicted Class 2: ",cluste
r_2.shape[0])
    print("Data points in Predicted Class 4: ",cluste
r_4.shape[0],end="\n\n")

    error_cluster_2 = cluster_2.loc[cluster_2['Class'
]== 4]
    error_cluster_4 = cluster_4.loc[cluster_4['Class'
]== 2]
    print("Error data points, Predicted Class 2:\n",e
rror_cluster_2,end="\n\n")
    print("Error data points, Predicted Class 4:\n",e
rror_cluster_4,end="\n\n")

    print("Number of all data points:   ", predicted_
clusters.shape[0],end="\n\n")

    print("Number of error data points: ", error_clus
ter_2.shape[0] + error_cluster_4.shape[0],end="\n\n")
```

```
    print("Error rate for class 2:       ", round((err
or_cluster_2.shape[0]/predicted_clusters.shape[0])*10
0,1),"%")
    print("Error rate for class 4:       ", round((err
or_cluster_4.shape[0]/predicted_clusters.shape[0])*10
0,1),"%")
    print("Total error rate:             ", round(((er
ror_cluster_2.shape[0] + error_cluster_4.shape[0])/pr
edicted_clusters.shape[0])*100,1),"%")


#invoke main method
if __name__ == "__main__":
    main()
```

```
Randomly selected row  559 for centroid m
u_2

Initial centroid mu_2:
A2     5
A3     1
A4     1
A5     1
A6     2
A7     1
A8     2
A9     1
A10    1
Name: 559, dtype: float64

Randomly selected row  447 for centroid m
u_4

Initial centroid mu_4:
A2     5
A3     1
```

```
A4      1
A5      1
A6      2
A7      1
A8      1
A9      1
A10     1
Name: 447, dtype: float64

Program ended after  5  iterations.

Final centroid mu_2:
 A2       7
A3        7
A4        7
A5        6
A6        5
A7        8
A8        6
A9        6
Class     4
```

```
dtype: float64

Final centroid mu_4:
 A2        3
A3         1
A4         1
A5         1
A6         2
A7         1
A8         2
A9         1
Class      2
dtype: float64

Final cluster assignment:

       Scn  Class  predicted_class
0   1e+06  2       4
1   1e+06  2       2
2   1e+06  2       4
3   1e+06  2       2
```

```
4    1e+06  2       4
5    1e+06  4       2
6    1e+06  2       4
7    1e+06  2       4
8    1e+06  2       4
9    1e+06  2       4
10   1e+06  2       4
11   1e+06  2       4
12   1e+06  4       4
13   1e+06  2       4
14   1e+06  4       2
15   1e+06  4       2
16   1e+06  2       4
17   1e+06  2       4
18   1e+06  4       2
19   1e+06  2       4

Data points in Predicted Class 2:  236
Data points in Predicted Class 4:  463

Error data points, Predicted Class 2:
```

```
        Scn  Class  predicted_class
5     1e+06  4       2
14    1e+06  4       2
15    1e+06  4       2
18    1e+06  4       2
20    1e+06  4       2
..      ...  ..      ..
681   1e+06  4       2
691   7e+05  4       2
696   9e+05  4       2
697   9e+05  4       2
698   9e+05  4       2

[225 rows x 3 columns]

Error data points, Predicted Class 4:
        Scn  Class  predicted_class
0     1e+06  2       4
2     1e+06  2       4
4     1e+06  2       4
6     1e+06  2       4
```

```
7    1e+06  2       4
..     ...  ..      ..
690  7e+05  2       4
692  7e+05  2       4
693  8e+05  2       4
694  8e+05  2       4
695  8e+05  2       4

[447 rows x 3 columns]

Number of all data points:    699

Number of error data points:  672

Error rate for class 2:       32.2 %
Error rate for class 4:       63.9 %
Total error rate:             96.1 %
```

In [ ]: